(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété Intellectuelle
Bureau international

(43) Date de la publication internationale
3 janvier 2002 (03.01.2002)

PCT

(10) Numéro de publication internationale
WO 02/01346 A2

(51) Classification internationale des brevets[7] : G06F 9/00

(21) Numéro de la demande internationale :
PCT/FR01/02019

(22) Date de dépôt international : 26 juin 2001 (26.06.2001)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
1015579 30 juin 2000 (30.06.2000) NL

(71) Déposant *(pour tous les États désignés sauf US)* : **THALES NEDERLAND B.V.** [NL/NL]; Zuidelijke Havenweg 40, NL-7554 RR Hengelo (NL).

(72) Inventeur; et

(75) Inventeur/Déposant *(pour US seulement)* : **BELTMAN, E., W.** [FR/FR]; Thales Intellectual Property, 13, av. du Prés. Salvador Allende, F-94117 Arcueil Cédex (FR).

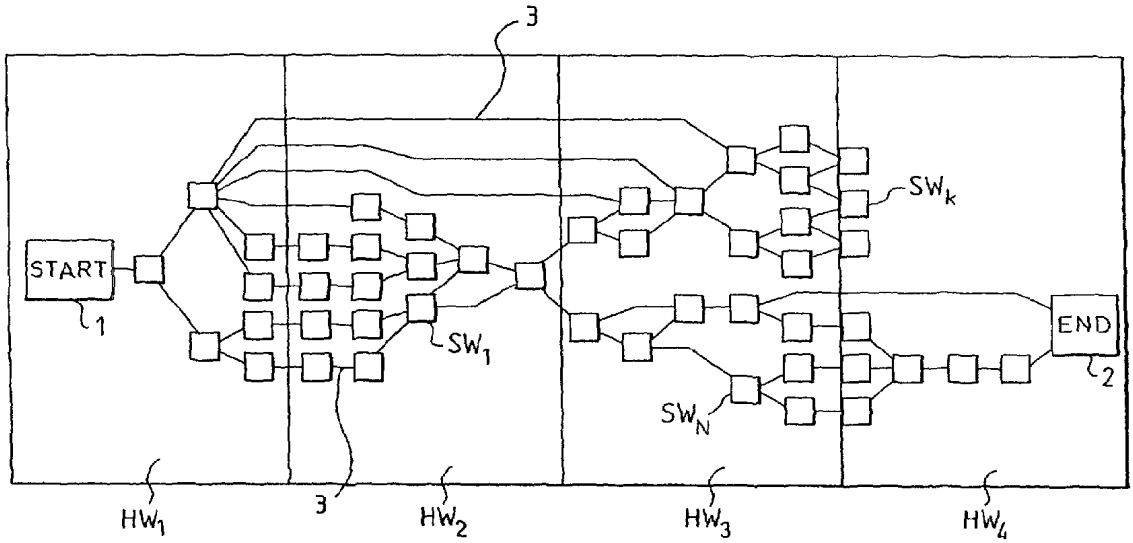
(74) Mandataire : **LUCAS, Laurent**; Thales Intellectual Property, 13, av. du Prés. Salvador Allende, F-94117 Arcueil Cédex (FR).

(81) États désignés *(national)* : AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

[Suite sur la page suivante]

(54) **Title:** METHOD FOR AUTOMATICALLY IMPLANTING SOFTWARE FUNCTIONS ON A SET OF PROCESSORS

(54) **Titre :** PROCEDE D'IMPLANTATION AUTOMATIQUE DE FONCTIONS LOGICIELLES SUR UNE ENSEMBLE DE PROCESSEURS



(57) **Abstract:** The invention concerns a method for automatically implanting software functions on a set of processors. The method comprises at least: a step which consists in breaking down the software functions into elementary tasks (SW_1, SW_k, , SW_N); a step which consists in implanting the elementary tasks on the processors (HW_1, HW_2, HW_3, HW_4); a step which consists in controlling implantation evaluating parameters whereof a list is pre-established, a penalty being assigned when a parameter does not fulfil a given criterion; a step which consists in calculating implantation cost, said cost being the sum of penalties assigned, the selected implantation being based on said cost. The invention is particularly applicable for systems, such as for example radar systems, comprising a large amount of software.

[Suite sur la page suivante]

WO 02/01346 A2

(84) États désignés *(régional)* **:** brevet européen (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

Déclaration en vertu de la règle 4.17 :

— *relative au droit du déposant de revendiquer la priorité de la demande antérieure (règle 4.17.iii)) pour toutes les désignations*

Publiée :

— *sans rapport de recherche internationale, sera republiée dès réception de ce rapport*

En ce qui concerne les codes à deux lettres et autres abréviations, se référer aux "Notes explicatives relatives aux codes et abréviations" figurant au début de chaque numéro ordinaire de la Gazette du PCT.

(57) Abrégé : La présente invention concerne un procédé d'implantation automatique de fonctions logicielles sur un ensemble de processeurs. Le procédé comporte au moins; une étape de décomposition des fonctions logicielles en tâches élémentaires (SW_1, ... SW_k, ... SW_N); une étape d'implantation des tâches élémentaires sur les processeurs (HW_1, HW_2, HW_3, HW_4); une étape de contrôle de paramètres d'évaluation de l'implantation dont une liste est préétablie, une pénalité étant attribuée à l'implantation lorsqu'un paramètre ne répond pas à un critère donné; une étape de calcul de coût de l'implantation, ce coût étant la somme des pénalités attribuées, l'implantation retenue étant fonction de ce coût. L'invention s'applique notamment pour des systèmes, tels que par exemple des systèmes radar, comportant une grande quantité de logiciels.

Procédé d'implantation automatique de fonctions logicielles sur un ensemble de processeurs

La présente invention concerne un procédé d'implantation automatique de fonctions logicielles sur un ensemble de processeurs. Elle s'applique notamment pour des systèmes, tels que par exemple des systèmes radar, comportant une grande quantité de logiciels.

La quantité de logiciels utilisés notamment dans les systèmes radar augmentent rapidement. De même, la quantité de processeurs utilisés augmente. Ces processeurs sont par exemple du type traitement du signal. Une application donnée peut nécessiter jusqu'à plusieurs dizaines de processeurs. Le travail d'implantation du logiciel sur les processeurs disponibles devient de plus en plus long et difficile. De plus, si des fonctions logicielles sont ajoutées ultérieurement, ce qui est fréquemment le cas, il est impossible de répartir de nouveau le logiciel sur l'ensemble des processeurs sans modification majeur de l'architecture logicielle ou matérielle.

Il apparaît donc que dans de tels systèmes, l'implantation des fonctions logicielles sur l'ensemble des processeurs est un problème crucial. Il y a certes un problème de temps et de complexité de mise en place des logiciels mais il y a aussi un problème de flexibilité. Il faut en effet pouvoir intégrer facilement des fonctions logicielles nouvelles. Ces problèmes entraînent des surcoûts de réalisation des systèmes. Par ailleurs, ils influent aussi notamment sur la maintenance, le test et la fiabilité de ces systèmes.

Un but de l'invention est notamment de permettre une implantation automatique et optimale du logiciel dans les processeurs, c'est-à-dire simple et économique. A cet effet, l'invention a pour objet un procédé d'implantation de fonctions logicielles sur un ensemble de processeurs, comportant au moins :

- une étape de décomposition des fonctions logicielles en tâches élémentaires et de création de fichiers définissant les liens entre les tâches et les connexions des processeurs ;

- une étape d'implantation des tâches élémentaires sur les processeurs en fonction des fichiers précédents ;
- une étape de contrôle de paramètres d'évaluation de l'implantation dont une liste est préétablie, une pénalité étant attribuée à l'implantation lorsqu'un paramètre ne répond pas à un critère donné ;
- une étape de calcul de coût de l'implantation, ce coût étant la somme des pénalités attribuées, l'implantation retenue étant fonction de ce coût.

L'implantation retenue correspond de préférence au coût minimal. Les étapes d'implantation des tâches, de contrôle des paramètres d'évaluation et de calcul de coût sont répétées, une implantation étant alors retenue lorsque la variation de coût converge en deçà d'un seuil donné, de l'ordre de 2% à 3% par exemple. Les paramètres d'évaluations sont notamment relatifs au flux de données, à la charge des processeurs et au temps de traitement des tâches.

L'invention a pour principaux avantages qu'elle permet une grande flexibilité d'implantation des différentes tâches d'un système, qu'elle augmente la fiabilité du système, qu'elle facilite la maintenance du système, et qu'elle facilite la sous-traitance de sous-ensembles.

D'autres caractéristiques et avantages de l'invention apparaîtront à l'aide de la description qui suit faite en regard de dessins annexés qui représentent :

- la figure 1, un exemple d'affectation de fonctions logicielles à un ensemble de processeurs ;
- la figure 2, une architecture logicielle correspondant schématiquement à l'affectation précédente ;
- la figure 3, de façon schématique une architecture logicielle obtenue en appliquant le procédé selon l'invention ;
- la figure 4, un exemple de moyen de contrôle de paramètres relatifs au flux de données entre les processeurs ;

- la figure 5, un exemple d'implantation de logiciels obtenue en appliquant le procédé selon l'invention.

La figure 1 illustre un exemple d'affectation (mapping) de N fonctions logicielles SW'_1, ... SW'_k, ... SW'_N à un ensemble de processeurs HW_1, HW_2, HW_3, HW_4. A titre d'exemple le nombre de processeurs est quatre. Certaines applications peuvent cependant nécessiter plusieurs dizaines de processeurs. Les fonctions logicielles regroupent des tâches d'un programme complet, par exemple un programme de traitement ou de simulation radar. Ce programme part d'une tâche de début 1 jusqu'à une tâche de fin 2. Le programme global peut par exemple comporter plusieurs centaines de tâches représentant au total plusieurs dizaines de milliers de lignes de code. Un processeur exécute une tâche SW'_k à la fois. Les lignes 3 de la figure 1 reliant les composants logiciels entre eux illustrent le déroulement ou l'interfaçage des tâches. Une ligne 3 qui relie deux composants logiciels indique que les deux tâches qu'ils exécutent peuvent se suivre. C'est-à-dire qu'une tâche est exécutée lorsque la précédente est achevée.

Les processeurs HW_1, HW_2, HW_3, HW_4 comportent notamment, outre les circuits de traitement proprement dit, les mémoires de programme et les circuits d'interface vers les autres composants matériels. Un processeur HW_k peut par exemple occuper une carte.

La figure 1 qui illustre par ailleurs le flux de données entre la tâche de début 1 et la tâche de fin 2 montre que l'attribution des composants logiciels n'est pas optimale. Un premier inconvénient est que l'attribution des tâches SW'_k n'est pas compatible avec leur interfaçage. A titre d'exemple, deux tâches éloignées l'une de l'autre dans le déroulement global des tâches sont exécutées par un même processeur. Etant donné qu'un processeur ne peut exécuter qu'une tâche à la fois, le temps de traitement global entre la tâche de début 1 et la tâche de fin 2 n'est pas optimisé. En effet, les aller-retour d'un processeur à l'autre dans le déroulement de tâches rallonge le temps de traitement. Un autre inconvénient est que le système est peu ou même pas flexible. Un changement d'algorithme peut entraîner une nouvelle

et complète répartition du logiciel, voire une modification de l'architecture matérielle.

La figure 1 illustre en fait le déroulement de l'exécution des différentes tâches d'un programme par quatre processeurs sans règle préétablie. Les parties de programme sont implantées essentiellement en fonction des disponibilités des processeurs. Les tâches élémentaires existent de fait, mais le programme n'est pas décomposé en tâches élémentaires de façon à répartir ces dernières parmi les processeurs. Certaines tâches peuvent même être à cheval sur deux processeurs. Cette attribution est complexe à mettre en œuvre et comme indiqué précédemment, elle manque de flexibilité.

La figure 2 présente une architecture logicielle correspondant à l'affectation des composants logiciels selon la figure 1. Pour chaque processeur HW_1, HW_2, ... HW_4 sont représentées les couches logicielles associées. Un système d'exploitation temps réel RTOS (real time operating system) est implanté sur chaque processeur. Ce système d'exploitation permet classiquement l'exécution du code 21, 22, 23 de programme. Ce dernier découle de spécifications 24. La couche logicielle définies par les codes de l'applicatif 21, 22, 23 comporte l'ensemble des tâches SW'_1, ... SW'_k, ... SW'_N précitées.

La figure 3 illustre de façon schématique une architecture logicielles obtenue en appliquant le procédé selon l'invention. Dans une première étape, le procédé selon l'invention décompose donc le programme en tâches élémentaires. Ces tâches sont programmées par des groupements de codes constituant des composants logiciels. Pour simplifier, une tâche élémentaire pourra être assimilée par la suite à son composant logiciel correspondant Cette décomposition est par exemple effectuée par un système de génie logiciel 31, encore appelé CASE, acronyme de l'expression anglo-saxonne « Computer-Assisted Software Engineering ». Cet outil CASE va par ailleurs définir la structure du logiciel, c'est-à-dire définir le lien entre les différentes tâches élémentaires ou la façon dont ces dernières dépendent les unes des autres. De préférence, la décomposition

est telle que les tâches élémentaires correspondent à des composants logiciels les plus petits possibles, c'est-à-dire ayant un nombre minimum de lignes de code, par exemple de l'ordre de 100 à 200. L'outil CASE réalise notamment cette structure en fonction des spécifications de départ 21. Cet outil définit par exemple une liste des tâches élémentaires décrivant par ailleurs la façon dont ces dernières sont connectées entre elles. Ces informations de structure logicielle et cette liste des tâches sont mémorisées dans un fichier 32. Les spécifications définissent par ailleurs la ou les fonctions du logiciel, mémorisées dans un fichier 33. De plus, une liste des processeurs disponibles décrivant la façon dont ces derniers sont connectés entre eux peut être établie. Cette liste définit la structure matérielle supportant l'ensemble du programme constitué des tâches élémentaires. Elle peut être mémorisée dans un fichier 34. Ces fichiers 32, 33, 34 constituent une description du système, exploitée par la suite pour l'implantation des tâches élémentaires.

Une fois la structure logicielle définie, ses composants logiciels sont implantés dans les différents processeurs. Cette affectation est par exemple réalisée par un deuxième outil logiciel 35, appelé outil DRAM par la suite, selon l'expression anglo-saxonne « Dependency Related Allocation and Mapping ». Cet outil effectue une première affectation des composants logiciels en fonction des fichiers 32, 33, 34 précédents. Cette affectation est par exemple réalisée de façon aléatoire. Puis l'outil DRAM réalise une série de contrôles de paramètres d'évaluation de l'implantation, sur des critères donnés. Ces critères sont par exemple relatifs au flux de données, à la charge des processeurs, au temps de traitement ou encore aux contraintes de conception. Lorsqu'un paramètre contrôlé ne répond pas à un critère donné, une pénalité est attribuée à l'implantation. Lorsque tous les paramètres, dont la liste est préétablie, sont contrôlés, l'outil DRAM calcule un coût qui est la somme des pénalités. L'implantation optimale est celle qui présente le coût minimal. En pratique, l'implantation retenue peut avoir un coût différent de ce coût minimal. En tout état de cause elle dépend du coût, une solution à coût trop élevée étant écartée.

Les lignes qui suivent décrivent un exemple de mise en œuvre possible du contrôle de paramètres caractérisant l'affectation des composants logiciels et des pénalités associées. Dans cet exemple, on considère que le système comporte quatre cartes HW_1, HW_2, HW_3, HW_4, chaque carte pouvant contenir un ou plusieurs processeurs.

La figure 4 illustre un moyen de contrôle des paramètres relatifs au flux de données. Pour le contrôle du flux de données, on effectue un traitement en pipeline à partir de requêtes entrantes sur la première carte HW_1. Plus précisément, une requête 41 de type « trigger » est envoyée en entrée de cette première carte. Cette requête entraîne l'activation de paramètres 42 en sortie de la quatrième carte HW_4. Cela implique par ailleurs que les paramètres traités par la première carte HW_1 activés par suite de la requête 41, soient traités par toutes les autres cartes HW_2, HW_3, HW_4 durant tout le traitement. Afin de minimiser les transferts de données non nécessaires, les données doivent être traitées juste à temps avant d'être utilisées. Etant donné qu'il est vraisemblable que toutes les données traitées, principalement les données issues directement de la requête 41, sont nécessaires aux autres cartes HW_2, HW_3, HW_4, la première carte HW_1 comporte des liaisons de communication 43, 44, 45 vers les autres processeurs. Pour chaque tâche élémentaire SW_k, l'outil DRAM piste le ou les tâches élémentaires produisant les données consommées par cette tâche élémentaire SW_k. La ou les tâches élémentaires produisant ces données peuvent être traitées par la même ou par une autre carte, et dans une même carte par un même ou par un autre processeur.

Un premier paramètre caractérisant le flux de données peut être la communication « forward » inter-processeurs. Dans ce cas, les données d'entrée d'une tâche élémentaire considérée sont traitées par la carte physiquement juste devant la carte traitant cette tâche. Par exemple, une tâche élémentaire de la troisième carte HW_3 nécessite un paramètre produit par la deuxième carte HW_2. Dans le but de minimiser les transferts de données, tous les paramètres d'entrée sont de préférence traités par le même processeur, ou au moins par la même carte. La pénalité donnée pour une communication forward inter-cartes serait plus forte qu'une

communication inter-processeurs. Il est encore plus pénalisant que des données transitent par deux cartes ou plus. Ainsi, l'outil DRAM peut par exemple multiplier la pénalité donnée pour une communication forward inter-cartes par le nombre de cartes parcourues entre la production des paramètres et leur utilisation. A titre d'exemple, une pénalité pour une communication forward inter-cartes peut être égale à 4.

Un deuxième critère relatif au flux de données peut être la communication backward inter-cartes. Le paramètre d'entrée est traité par une autre carte. Cette carte est physiquement derrière la carte courante, c'est-à-dire que les données ne sont pas encore traitées. Par exemple, une tâche élémentaire de la deuxième carte HW_2 nécessite un paramètre produit par une tâche élémentaire produite par la carte HW_3. Pour minimiser le transfert de données d'une carte à l'autre, les données doivent circuler dans une seule direction, de la première carte HW_1 vers la dernière carte HW_4. Si l'on considère qu'il est important d'éviter les communications backward inter-cartes, une forte pénalité peut être attribuée à une telle communication. Cette pénalité peut être par exemple égale à 1000.

Un autre type de critère à prendre en compte est la charge de traitement des processeurs. L'outil DRAM prend en compte les charges des processeurs dans le but notamment de ne pas placer trop de tâches dans un même processeur. Pour chaque tâche élémentaire, la charge de processeur nécessitée est définie dans un fichier d'entrée, qui est par exemple le même fichier que celui contenant la liste de ces tâches. A titre d'exemple, la charge du processeur pour l'exécution d'une tâche élémentaire peut être définie pour l'utilisation maximale théorique, l'utilisation maximale pratique ou l'utilisation moyenne de cette tâche. De cette façon, la charge du processeur est utilisée comme une méthode directe pour assurer une affectation correcte des tâches parmi les processeurs.

Un premier cas à prendre en compte est celui où l'exécution d'une tâche entraîne le dépassement d'un seuil, par exemple de 95% de la charge maximum autorisée du processeur. L'outil DRAM ne doit pas autoriser ce cas. La pénalité pour le dépassement de ces 95% peut alors être égale à

10000. Il est par ailleurs possible d'envisager plusieurs niveaux de surcharges en dessous du maximum absolu de 95% avec des pénalités décroissantes. Une pénalité peut par exemple être aussi attribuée si un processeur est chargé insuffisamment. Cela incite notamment à utiliser au maximum les processeurs disponibles, dans la limite bien sûr des surcharges admissibles.

Un autre critère important à prendre en compte est le temps de traitement des données. Le temps de traitement peut être considéré de deux façons au moins. Un premier temps d'exécution à contrôler peut être le temps nécessaire pour exécuter toutes les tâches affectées à un processeur avec leurs fréquences d'exécution. Cette fréquence d'exécution est définie pour chaque tâche, les informations étant par exemple stockées dans le fichier de description des tâches. L'outil DRAM doit par exemple vérifier que ce temps total d'exécution des tâches sur un même processeur ne dépasse pas une durée donnée. Cette vérification peut être nécessaire, car il n'existe pas obligatoirement une relation entre la charge du processeur et ce temps d'exécution. A titre d'exemple, lorsque pour un processeur ce temps d'exécution dépasse une valeur donnée, la pénalité attribuée peut être égale à 10000. Il est possible d'attribuer des pénalités décroissantes en fonction de seuils de temps d'exécution décroissants.

Un deuxième temps d'exécution à prendre en compte, est le temps d'exécution du programme complet. Toutes les branches d'exécution de programme sont traitées par l'outil DRAM sur chaque carte. La branche ayant le plus long temps cumulé d'exécution de programme détermine le temps de traitement relatif à la carte. Le temps de traitement du programme par le système complet est la somme des temps de traitement de chaque carte HW_1, HW_2, HW_3, HW_4. Un défaut de respect d'un temps de traitement maximum autorisé peut être sévèrement sanctionné par exemple par une pénalité atteignant 250000. Diminuer le temps de traitement autorisé, ou augmenter la pénalité pour excès de temps de traitement, incite l'outil DRAM à faire fonctionner les processeurs en parallèle.

Une autre série de critères à contrôler peut concerner les contraintes de conception du système. Pour différentes raisons, les concepteurs peuvent vouloir influencer le mapping, c'est-à-dire l'implantation des tâches élémentaires dans les processeurs. Le procédé selon l'invention peut permettre plusieurs facilités d'implantation. En particulier, le concepteur peut imposer le placement d'une tâche sur un processeur spécifique, cette affectation étant par exemple précisée dans le fichier de description du système. Etant donné que l'outil DRAM n'intervient pas sur cette affectation, il n'y a pas lieu d'attribuer de pénalités. Cependant, si une situation dangereuse apparaît concernant par exemple le temps de traitement ou la charge du processeur, l'outil DRAM peut envoyer un message d'alerte.

Il est encore possible de coupler plusieurs tâches élémentaires, c'est-à-dire d'imposer de les affecter sur un même processeur ou une même carte, ce processeur ou cette carte n'étant pas imposé. Cela peut notamment être intéressant lorsque les tâches partagent le même pool de données. Ce couplage des tâches peut être précisé dans le fichier de description. Le non-respect de cette contrainte peut être pénalisé par une relativement forte pénalité, par exemple égale à 100. Si le couplage est impossible par suite de violation de règles de conception, concernant par exemple le temps de traitement ou la charge des processeurs, l'outil DRAM envoie un message d'alerte. L'ensemble des pénalités est par exemple défini dans le fichier de configuration de l'outil DRAM.

Les pénalités sont de préférence choisies avec précaution. En particulier, il faut veiller à faire correspondre le degré de pénalité avec le degré de contrainte attaché à un paramètre de conception. Les différentes règles de conception sont reliées les unes aux autres, en particulier elles s'influent les unes les autres. La meilleure conception peut être la meilleure combinaison de ces règles. Ainsi, la meilleure conception devrait être celle dont le coût exprimé sous forme de pénalités est le plus faible. Selon l'invention, une méthode pour déterminer le coût minimum ou du moins un coût s'en approchant, consiste à répéter l'algorithme d'implantation des tâches élémentaires. Lorsque l'on répète cet algorithme, les différentes

solutions obtenues ont des pénalités différentes. Les étapes répétées sont les suivantes :

- l'étape d'implantation des tâches élémentaires sur les processeurs ;
- l'étape de contrôle de paramètres d'évaluation de l'implantation dont une liste est préétablie;
- l'étape de calcul de coût de l'implantation.

On considère que l'on peut obtenir plusieurs bonnes solutions. Pour ces meilleures solutions, le coût en pénalité varie peu de l'une à l'autre. On prend par exemple un seuil compris entre 2% et 3%. Quand la variation de coût d'une itération à l'autre reste en deçà de ce seuil, on considère que la solution est acceptable. En pratique une solution peut donc être retenue lorsque la variation de coût converge en deçà du seuil, par exemple en deçà de 2% à 3% Pendant une première phase, toutes les tâches sont affectées à un processeur pris au hasard. Pendant les autres phases, d'une itération à l'autre, seulement une tâche prélevée au hasard est ré-affectée à un autre processeur, choisi lui aussi au hasard. Pour chacune de ces itérations, le coût est calculé.

La durée totale d'affectation des tâches (mapping) réalisée par l'outil DRAM peut prendre moins de 5 minutes pour une itération. Un grand nombre d'itérations peuvent donc être accomplies par cet outil. La meilleure implantation peut donc être obtenue relativement rapidement et automatiquement, donc de façon économique. Comme il a été indiqué précédemment, le coût en pénalités indique si cette implantation finale est obtenue. En se rapprochant d'une solution acceptable, le coût devrait varier de moins en moins d'une itération à l'autre, à condition néanmoins que les pénalités soient choisies avec précaution, c'est-à-dire en fonction du degré de contrainte attaché à un paramètre de conception. Si les coûts varient beaucoup, cela indique qu'un ou plusieurs paramètres d'affectation ne sont pas du tout corrects. Au lieu de commencer par un mapping complètement aléatoire, il est possible de prévoir une présélection de processeurs en fonction de certains critères, par exemples liés au flux de données.

Une fois l'affectation des tâches réalisées par l'outil DRAM, c'est-à-dire une fois qu'une solution a été déterminée avec un coût en pénalités acceptable, un générateur de code 36 crée un code qui permet à toutes ces tâches élémentaires d'être exécutées par les différents processeurs ou cartes HW_1, HW_2..., HW_4 en jeu. A une tâche élémentaire correspond un composant logiciel SW_k, comportant quelques dizaines ou quelques centaines de lignes de code. Les parties hachurées de la figure 3 correspondent au code généré par le générateur de code 36. Ce dernier produit une couche intermédiaire (middleware) 37 qui communique avec les systèmes d'exploitation (opérating system) RTOS des processeurs ou cartes. Le générateur de code produit également une couche logicielle 38 autour de chaque composant logiciel, lui permettant de communiquer avec la couche middleware 37 et donc d'être exécuté par le processeur correspondant. Le générateur crée donc en quelque sorte un code glu (« glue » code) qui lie les composants logiciel HW_k aux processeurs. Il permet en fait de connecter les composants logiciels aux emplacements physiques, processeurs et cartes, indiqués par l'outil DRAM, conformément à l'implantation retenue.

La figure 5 illustre une affectation possible de composants logiciels SW_1, ... SW_k,... SW_N aux cartes HW_1, HW_2, HW_3, HW_4 obtenue en appliquant le procédé selon l'invention. Cette figure montre que les contraintes liées au flux des données sont bien prises en compte. En particulier, les composants logiciels SW_k sont mieux ordonnés sur l'ensemble des cartes HW_1, HW_2, HW_3, HW_4. Les autres contraintes, notamment de charge et de temps d'exécution sont bien sûr respectées. La figure 5 illustre d'autres avantages de l'invention. Elle montre en particulier qu'une grande facilité de maintenance et une fiabilité accrues sont obtenues. En particulier, en cas de défaillance d'une carte, il est aisé de remplacer celle-ci sans problèmes d'interfaces avec les autres cartes. Il est aussi plus facile et plus économique de sous-traiter des sous-ensembles du système complet. Dans l'exemple d'une implantation telle que présentées par la figure 5, un premier sous-traitant peut prendre en charge la première carte, matériel et logiciel compris, un deuxième sous-traitant peut prendre en charge la deuxième carte et ainsi de suite. L'assemblage des cartes se fait ensuite sans

problème, l'interfaçage matériel et fonctionnel d'une carte à l'autre étant facile.

Enfin, l'invention permet une grande flexibilité d'implantation. En effet, en cas d'ajout d'une ou plusieurs tâches, il est aisé d'appliquer le procédé selon l'invention, en utilisant par exemple l'outil DRAM. Dans ce cas, on part par exemple de la configuration existante, en affectant la ou les nouvelles tâches au hasard. Puis on lance les itérations. Le résultat peut notamment imposer l'ajout d'un nouveau processeur si aucun coût acceptable n'est obtenu.

L'invention a été présentée pour un exemple d'application à quatre processeurs, mais le nombre de processeurs peut bien sûr être plus important.

REVENDICATIONS

1. Procédé d'implantation de fonctions logicielles sur un ensemble de processeurs, caractérisé en ce qu'il comporte au moins :

- une étape de décomposition des fonctions logicielles en tâches élémentaires (SW_1, ... SW_k, ... SW_N), de création d'un fichier (32) définissant le lien entre les différentes tâches élementaires et de création d'un fichier (34) définissant les connexions des processeurs (HW_1, HW_2, HW_3, HW_4) ;
- une étape d'implantation des tâches élémentaires sur les processeurs (HW_1, HW_2, HW_3, HW_4) en fonction des fichiers précédents (32, 34) ;
- une étape de contrôle de paramètres d'évaluation de l'implantation dont une liste est préétablie, une pénalité étant attribuée à l'implantation lorsqu'un paramètre ne répond pas à un critère d'évaluation donné ;
- une étape de calcul de coût de l'implantation, ce coût étant la somme des pénalités attribuées, l'implantation retenue étant fonction de ce coût.

2. Procédé selon la revendication 1, caractérisé en ce que l'implantation retenue correspond sensiblement au coût minimal.

3. Procédé selon l'une quelconque des revendications précédentes, caractérisé en ce que les étapes d'implantation des tâches, de contrôle des paramètres d'évaluation et de calcul de coût sont répétées, une implantation étant retenue lorsque la variation de coût converge en deçà d'un seuil donné.

4. Procédé selon la revendication 3, caractérisé en ce que lors de la première itération, toutes les tâches sont affectées à un processeur au hasard, puis seulement une tâche prélevée au hasard est ré-affectée à un autre processeur d'une itération à l'autre.

5. Procédé selon l'une quelconque des revendications précédentes, caractérisé en ce que les paramètres d'évaluations sont relatifs au flux de données, à la charge des processeurs et au temps de traitement des tâches.

6. Procédé selon l'une quelconque des revendications précédentes, caractérisé en ce qu'à une tâche élémentaire (SW_1, ... SW_k,... SW_N) correspondant un composant logiciel, un générateur de code (36) crée une couche intermédiaire (middleware) (37) qui communique avec les systèmes d'exploitation (opérating system, RTOS) des processeurs, le générateur de code produisant également une couche logicielle (38) autour de chaque composant logiciel, lui permettant de communiquer avec la couche intermédiaire (37) et donc d'être exécuté par le processeur correspondant, conformément à l'implantation retenue.

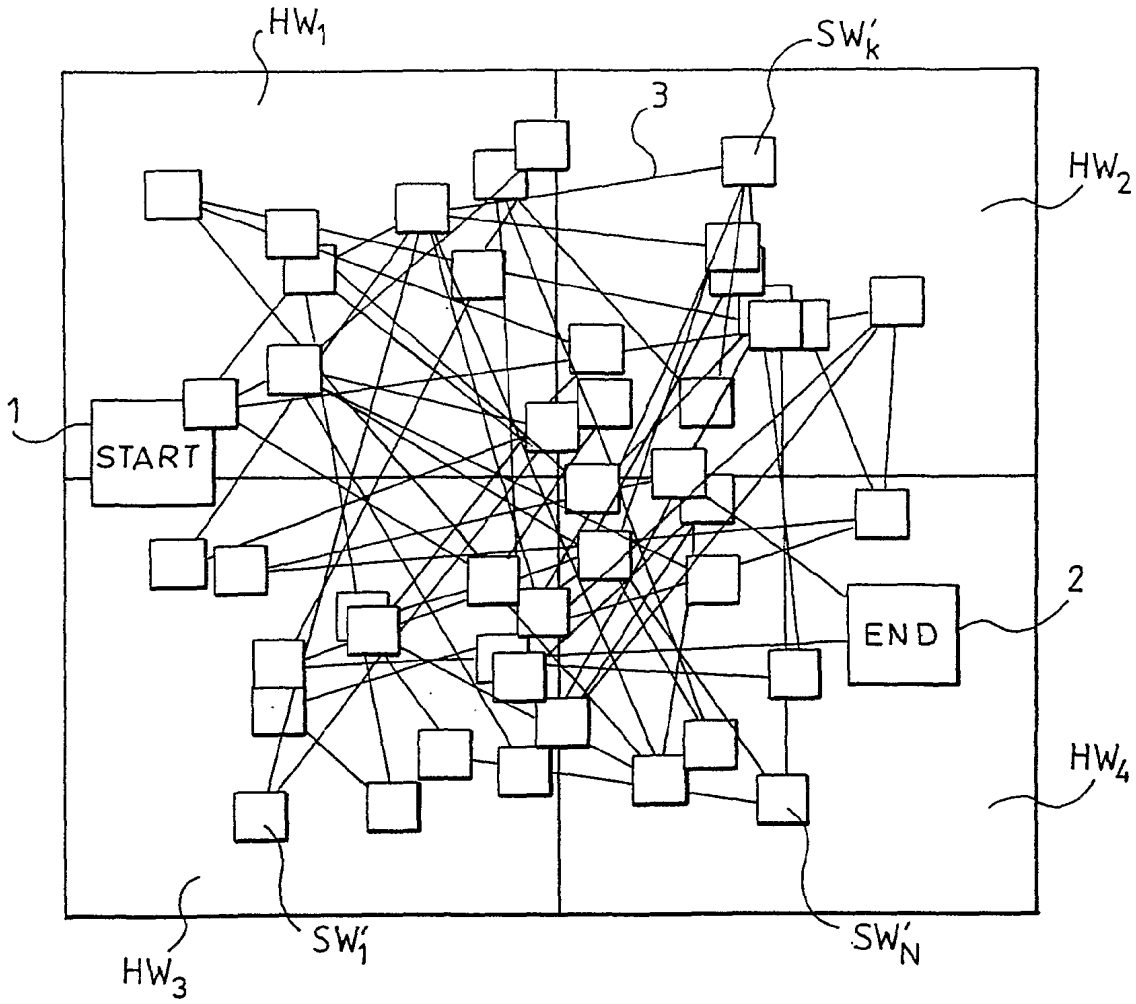


FIG.1

FIG. 2

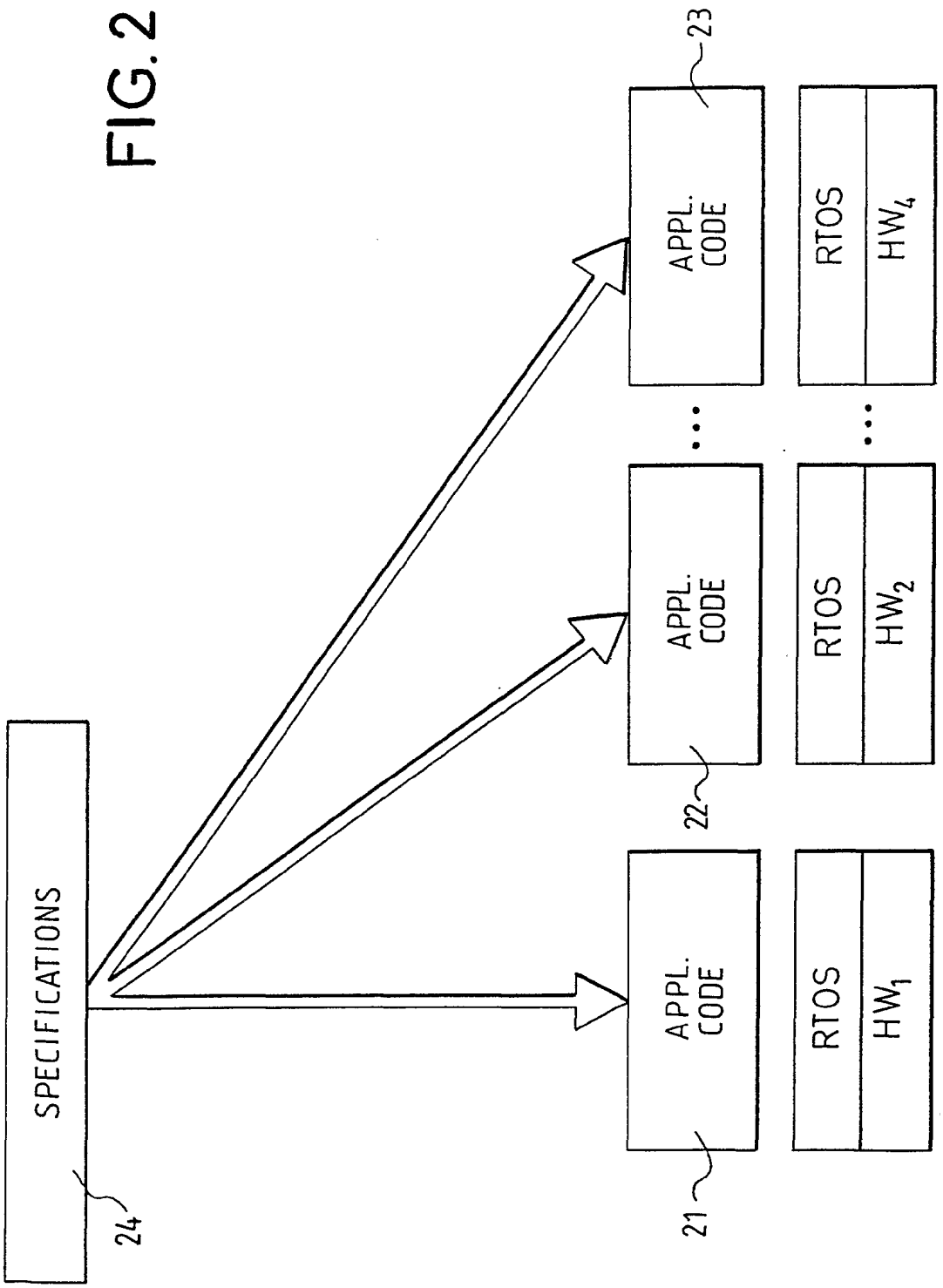
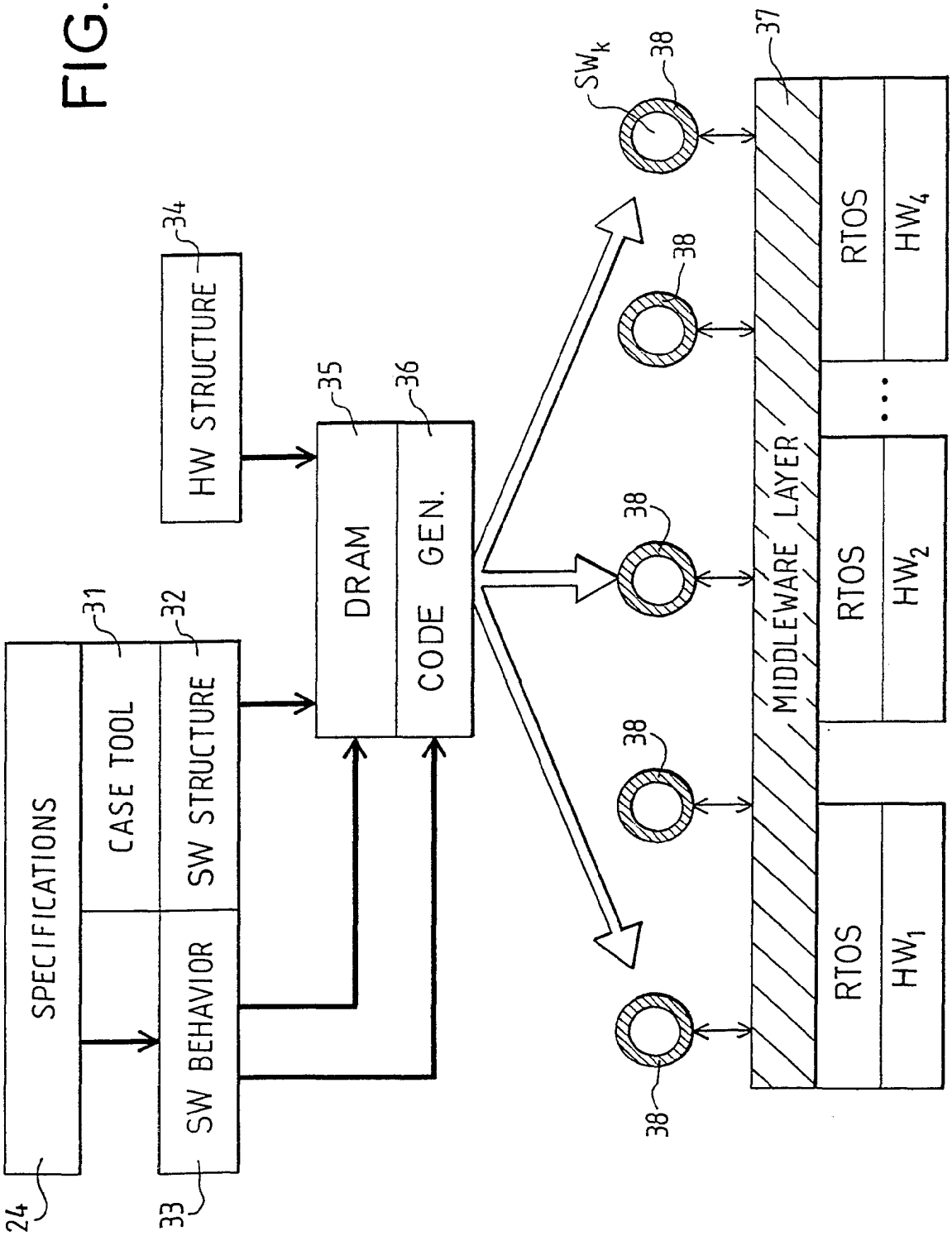

SPECIFICATIONS
24
APPL. CODE
21
RTOS
HW1
APPL. CODE
22
RTOS
HW2
APPL. CODE
23
RTOS
HW4

FIG.3


SPECIFICATIONS
CASE TOOL
SW STRUCTURE
SW BEHAVIOR
HW STRUCTURE
DRAM
CODE GEN.
MIDDLEWARE LAYER
RTOS
HW1
RTOS
HW2
RTOS
HW4
SWk
24
31
32
33
34
35
36
37
38

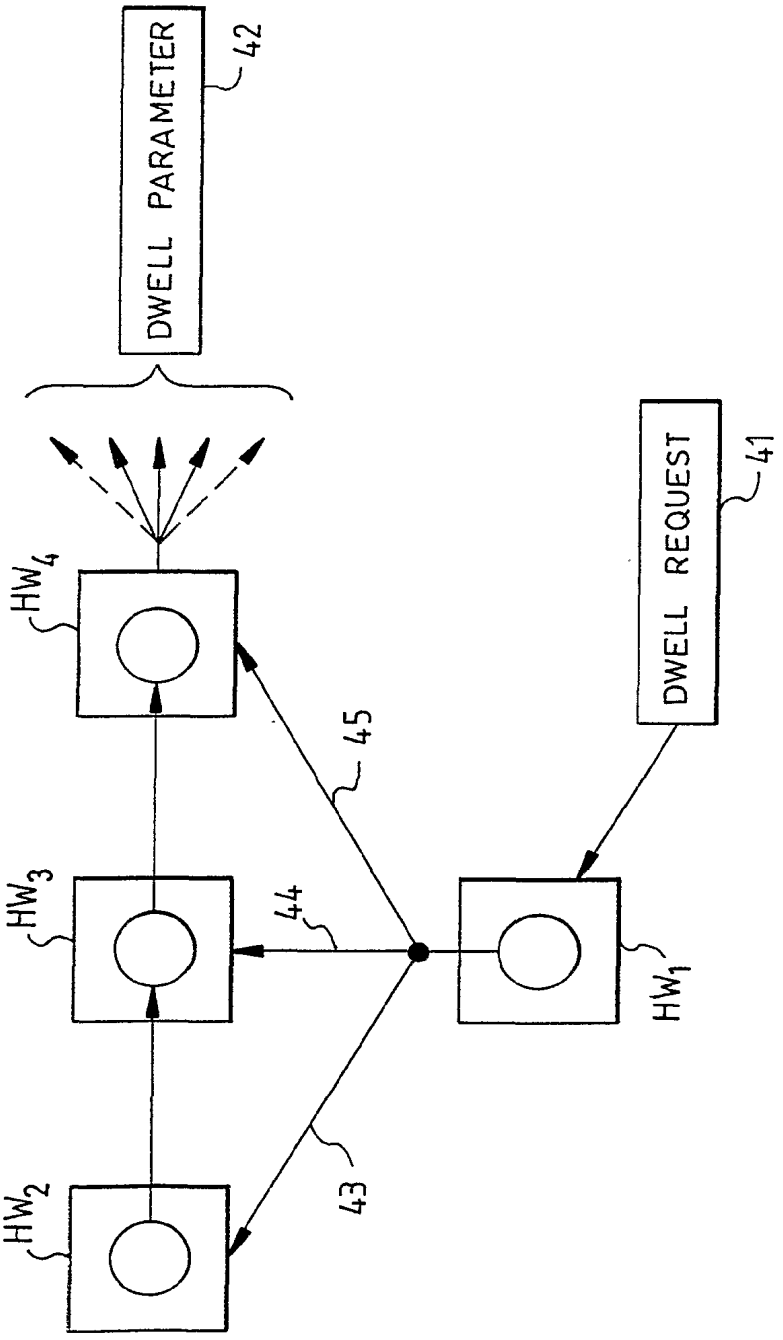


FIG.4

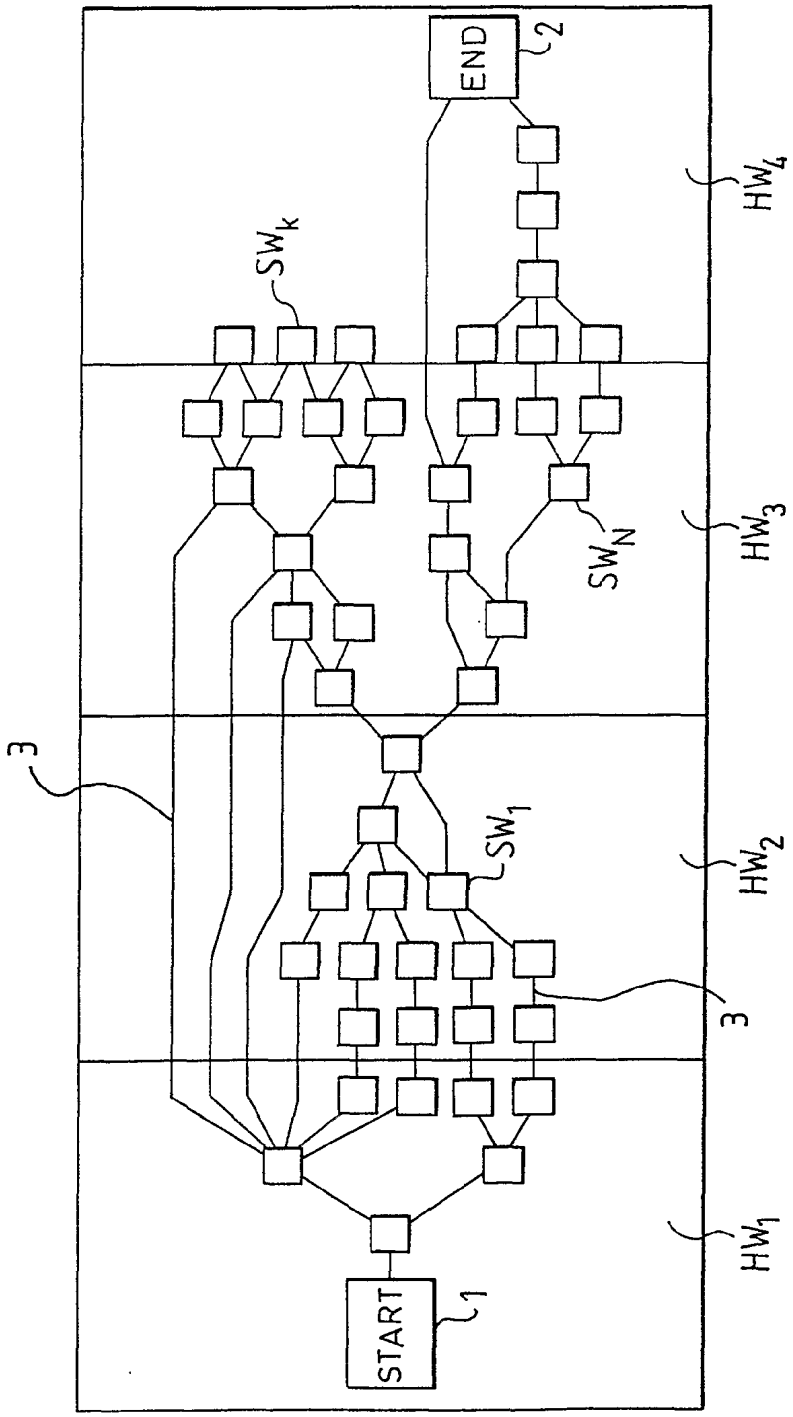


FIG.5